Name: Santos Tristan Neal U.

Section: CPE32S9

Date: 05-04-2024

- 1.Load time series data: data.csvDownload data.csv
- 2.Visualize the time series
- 3.Fit an ARIMA Model (baseline model order = (1,1,1))
- 4.Improve the ARIMA Model
- 5.Print the model summary
- 6.Make a forecast (steps=10)
- 7.Plot the forecast
- 8.Perform a grid search

+ Code + Text

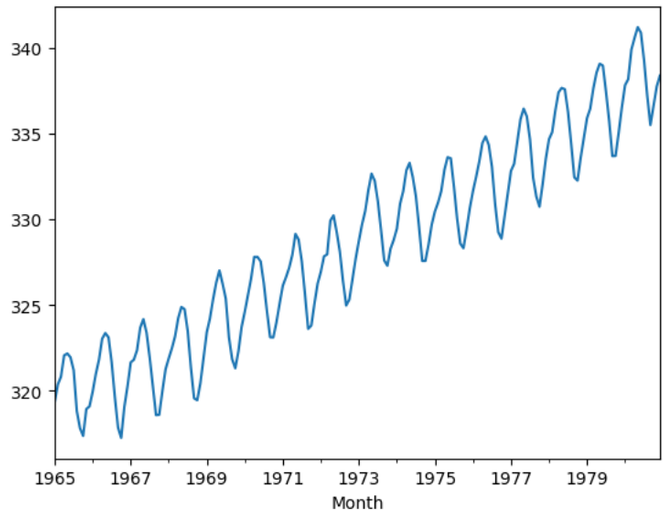
- 1.Load time series data: data.csvDownload data.csv

```
from google.colab import drive
drive.mount('/content/drive')
```

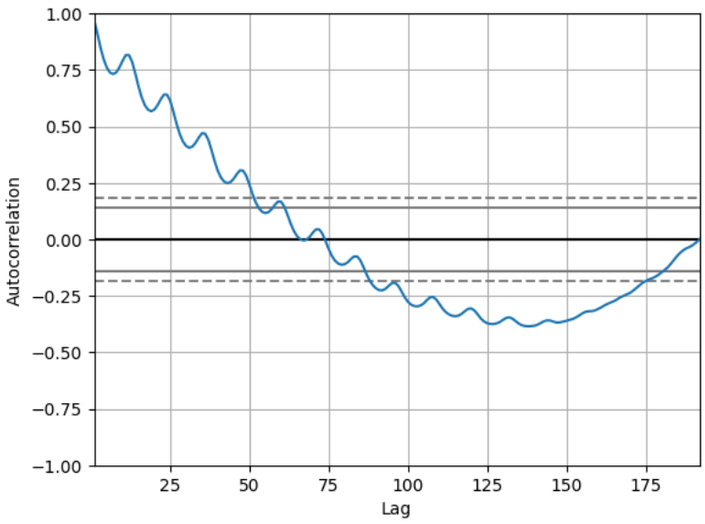
```
Mounted at /content/drive
```

```
from pandas import read_csv
from datetime import datetime
from matplotlib import pyplot

def parser(x):
    return datetime.strptime(x, '%Y-%m')

# load dataset
data = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/data.csv', header=0, parse_dates=[0], index_col=0, date_parser=parser)

# convert to Series
series = data.iloc[:, 0]

print(series.head())
series.plot()
pyplot.show()
```

```
Month
1965-01-01    319.32
1965-02-01    320.36
1965-03-01    320.82
1965-04-01    322.06
1965-05-01    322.17
Name: CO2 (ppm), dtype: float64
```




```
from pandas import read_csv
from datetime import datetime
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot

def parser(x):
    return datetime.strptime(x + '-01', '%Y-%m-%d')

# load dataset
series = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/data.csv', header=0, parse_dates=[0], index_col=0, date_parser=parser)

autocorrelation_plot(series)
pyplot.show()
```

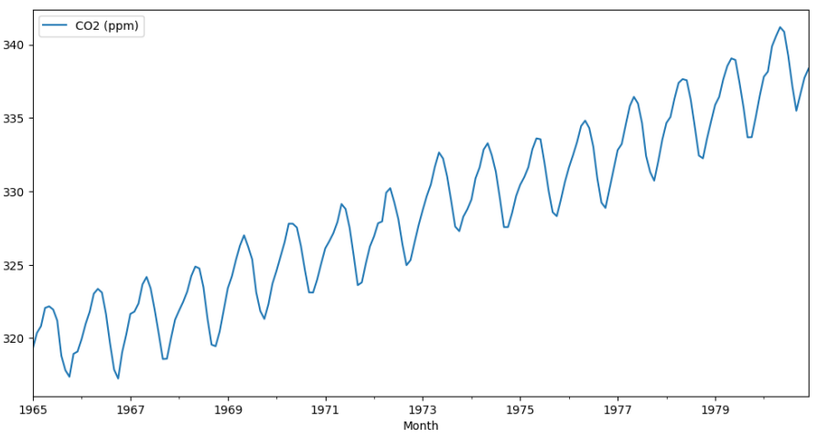



```
from pandas import read_csv
from statsmodels.tsa.arima.model import ARIMA

# load dataset
series = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/data.csv', header=0, index_col=0, parse_dates=True)

# fit model
model = ARIMA(series, order=(5,1,0))
model_fit = model.fit()
```

- 2.Visualize the time series

```
data.plot(figsize=(12,6))
```

```
<Axes: xlabel='Month'>
```



- 3.Fit an ARIMA Model (baseline model order = (1,1,1))

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(data, order=(1,1,1))
model_fit = model.fit()
```

- 4.Improve the ARIMA Model

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(data, order=(1,2,3))
model_fit = model.fit()
```

- 5.Print the model summary

```
print(model_fit.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:              CO2 (ppm)   No. Observations:                  192
Model:                 ARIMA(1, 2, 3)   Log Likelihood                -215.841
Date:                Sat, 04 May 2024   AIC                            441.682
Time:                        13:27:42   BIC                            457.917
Sample:                  01-01-1965   HQIC                           448.259
```

```
                           - 12-01-1980
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.3825      0.140      2.724      0.006       0.107       0.658
ma.L1         -0.4704     22.759     -0.021      0.984     -45.077      44.136
ma.L2         -0.1176     11.997     -0.010      0.992     -23.632      23.396
ma.L3         -0.4119      9.344     -0.044      0.965     -18.725      17.901
sigma2         0.5557     12.617      0.044      0.965     -24.172      25.284
===================================================================================
Ljung-Box (L1) (Q):                   0.37   Jarque-Bera (JB):                 2.16
Prob(Q):                              0.54   Prob(JB):                         0.34
Heteroskedasticity (H):               0.95   Skew:                             0.21
Prob(H) (two-sided):                  0.83   Kurtosis:                         2.68
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

- 6.Make a forecast (steps=10)

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt

# Splitting into train and test sets
train_size = int(len(data) * 0.75)
train_data, test_data = data[0:train_size], data[train_size:]

start = len(train_data)
end=len(train_data)+len(test_data)-1
model = ARIMA(train_data, order = (9,2,0))
results = model.fit()
predictions = results.predict(start=start, end=end, dynamic=False, type='levels')
forecast = results.predict(len(data), len(data)+10, typ = 'levels')


print("Forecast (steps = 10)")
print(forecast)
```

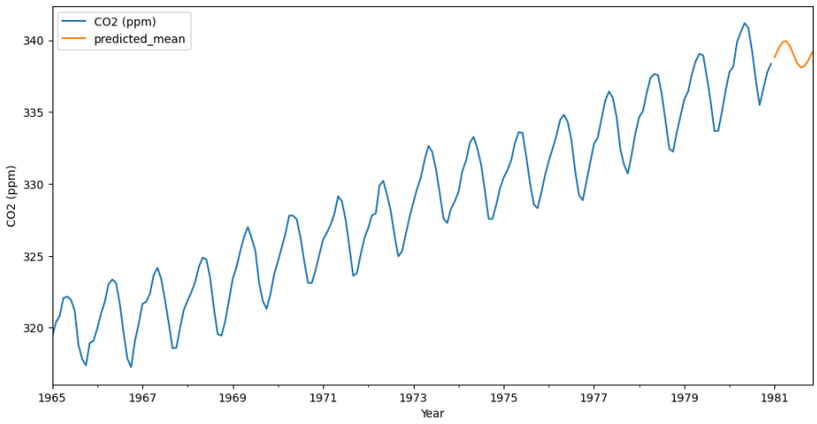
```
Forecast (steps = 10)
1981-01-01    338.829450
1981-02-01    339.410077
1981-03-01    339.846441
1981-04-01    339.951027
1981-05-01    339.624439
1981-06-01    338.996781
1981-07-01    338.387871
1981-08-01    338.097810
1981-09-01    338.219246
1981-10-01    338.635268
1981-11-01    339.168786
Freq: MS, Name: predicted_mean, dtype: float64
```

```
print("MSE: ", mean_squared_error(test_data, predictions))
print("RMSE: ", sqrt(mean_squared_error(test_data, predictions)))
```

```
MSE:  1.1498843083979555
RMSE:  1.0723265866320557
```

- 7.Plot the forecast

```
import matplotlib.ticker as ticker
formatter = ticker.StrMethodFormatter('{x:,.0f}')


ylabel='CO2 (ppm)'
xlabel='Year'

ax = data.plot(legend=True,figsize=(12,6))
forecast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter);
```

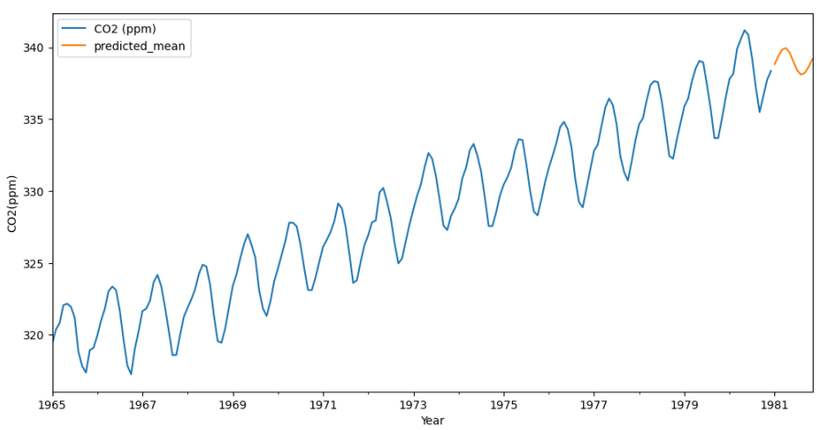


- 8.Perform a grid search

```
def evaluate_arima_model(dataset, arima_order):
  train_size = int(len(data) * 0.75)
  train_data, test_data = data[0:train_size], data[train_size:]

  start = len(train_data)
  end=len(train_data)+len(test_data)-1

  model = ARIMA(train_data, order = arima_order)
  results = model.fit()
  forecast = results.predict(len(data), len(data)+10, typ = 'levels')
  predictions = results.predict(start=start, end=end, dynamic=False, type='levels')
  mse = mean_squared_error(test_data, predictions)
  return mse


def evaluate_models(dataset, p_values, d_values, q_values):
    best_score, best_cfg = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p,d,q)
                try:
                    mse = evaluate_arima_model(data, order)
                    if mse < best_score:
                        best_score, best_cfg = mse, order
                    print('ARIMA%s MSE=%.3f' % (order,mse))
                except:
                    continue
    print('Best ARIMA=%s MSE=%.3f' % (best_cfg, best_score))


import warnings
p_values = range(0,11)
d_values = range(0, 4)
q_values = range(0, 4)
warnings.filterwarnings("ignore")
evaluate_models(data.values, p_values, d_values, q_values)
```

```
ARIMA(9, 2, 0) MSE=1.150
ARIMA(9, 2, 1) MSE=1.833
ARIMA(9, 2, 2) MSE=1.878
ARIMA(9, 2, 3) MSE=6.436
ARIMA(9, 3, 0) MSE=5080.474
ARIMA(9, 3, 1) MSE=1.468
ARIMA(9, 3, 2) MSE=36.215
ARIMA(9, 3, 3) MSE=7.989
ARIMA(10, 0, 0) MSE=18.554
ARIMA(10, 0, 1) MSE=18.698
ARIMA(10, 0, 2) MSE=18.717
ARIMA(10, 0, 3) MSE=17.485
ARIMA(10, 1, 0) MSE=17.642
ARIMA(10, 1, 1) MSE=17.734
ARIMA(10, 1, 2) MSE=17.729
ARIMA(10, 1, 3) MSE=17.150
ARIMA(10, 2, 0) MSE=1.906
ARIMA(10, 2, 1) MSE=3.473
ARIMA(10, 2, 2) MSE=3.965
ARIMA(10, 2, 3) MSE=3.705
ARIMA(10, 3, 0) MSE=1464.266
ARIMA(10, 3, 1) MSE=1.740
ARIMA(10, 3, 2) MSE=15.112
ARIMA(10, 3, 3) MSE=6.700
Best ARIMA=(9, 2, 0) MSE=1.150
```

```
# Splitting into train and test sets
train_size = int(len(data) * 0.75)
train_data, test_data = data[0:train_size], data[train_size:]

start = len(train_data)
end=len(train_data)+len(test_data)-1
model = ARIMA(train_data, order = (9,2,0))
results = model.fit()
predictions = results.predict(start=start, end=end, dynamic=False, type='levels')
forecast = results.predict(len(data), len(data)+10, typ = 'levels')

ylabel='CO2(ppm)'
xlabel='Year'

ax = data.plot(legend=True,figsize=(12,6))
forecast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter);
```



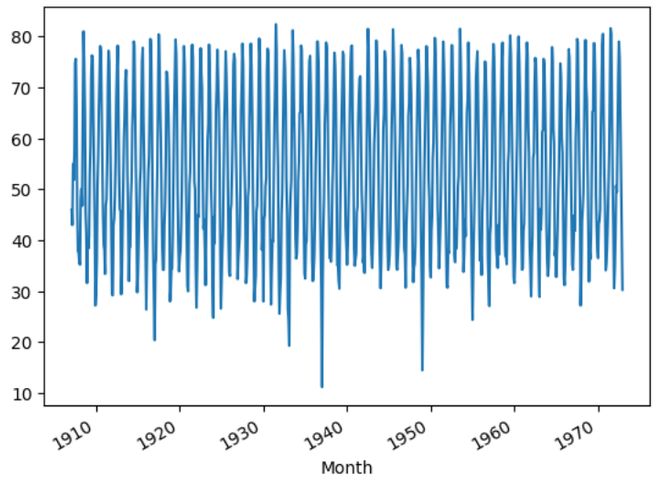
Supplementary

do the same for this dataset - dataset_temperature.csv

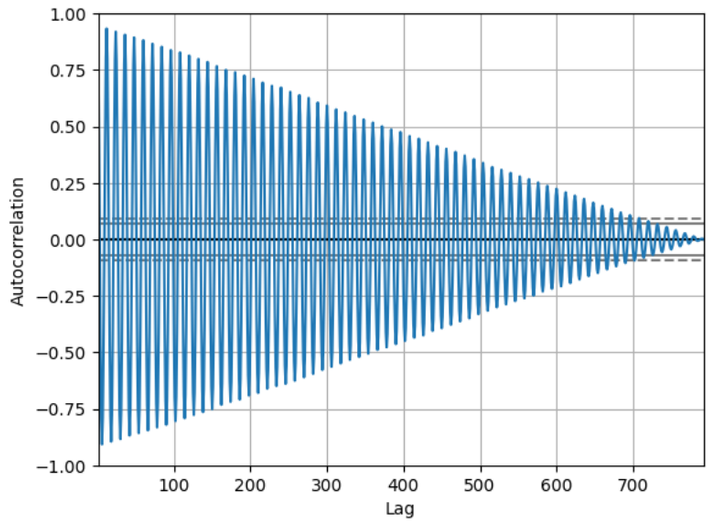
- 1.Load time series data: data.csvDownload dataset_temperature.csv

```
from pandas import read_csv
from datetime import datetime
from matplotlib import pyplot
import pandas as pd
import numpy as np

def parser(x):
    if x == ' 1907 ? 1972':
        return np.nan
    try:
        return datetime.strptime(str(x), '%Y-%m')
    except ValueError:
        return pd.NaT

# load dataset
data = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/dataset_temperature.csv', header=0, parse_dates=[0], index_col=0, date_p

# convert to Series
series = data.iloc[:, 0]

print(series.head())
series.plot()
pyplot.show()
```

```
Month
1907-02-01    46.0
1907-03-01    43.0
1907-04-01    55.0
1907-05-01    51.8
1907-06-01    57.5
Name: Mean monthly temperature, dtype: float64
```




```
from pandas import read_csv
from datetime import datetime
from matplotlib import pyplot
from pandas.plotting import autocorrelation_plot
import pandas as pd
import numpy as np

def parser(x):
    try:
        return datetime.strptime(x, '%Y-%m-%d')
    except (ValueError, TypeError):
        return np.nan

# Load dataset
df = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/dataset_temperature.csv')

# Convert 'Mean monthly temperature' column to numeric
df['Mean monthly temperature'] = pd.to_numeric(df['Mean monthly temperature'], errors='coerce')

# Remove rows with missing values
df.dropna(inplace=True)

# Extract the series to plot
series = df['Mean monthly temperature']

# Plot autocorrelation
autocorrelation_plot(series)
pyplot.show()
```

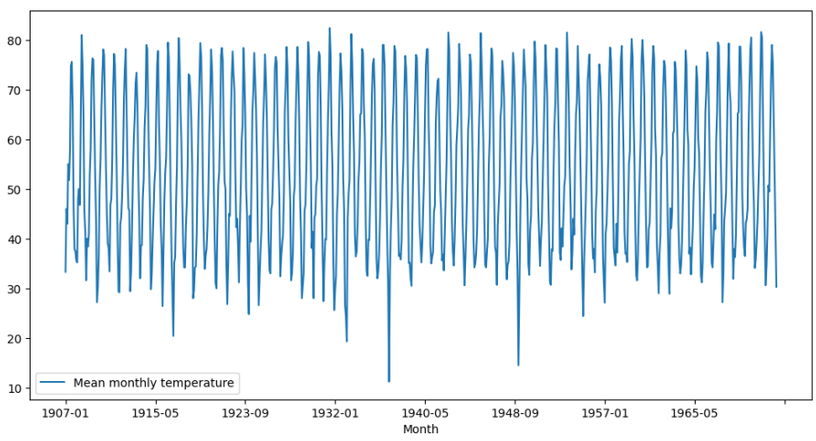



```
from pandas import read_csv
from statsmodels.tsa.arima.model import ARIMA
import pandas as pd

# Load dataset
series = read_csv('/content/drive/MyDrive/Colab Notebooks/Emtech2/csv/dataset_temperature.csv', header=0, index_col=0, parse_dates=True)

# Convert series to numeric format
series['Mean monthly temperature'] = pd.to_numeric(series['Mean monthly temperature'], errors='coerce')

# Remove rows with missing values
series.dropna(inplace=True)

# Fit model
model = ARIMA(series, order=(5, 1, 0))
model_fit = model.fit()
```

- 2.Visualize the time series

```
# Plot the dataset
series.plot(figsize=(12, 6))
```

```
<Axes: xlabel='Month'>
```



- 3.Fit an ARIMA Model (baseline model order = (1,1,1))

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(series, order=(1, 1, 1))
model_fit = model.fit()
```

- 4.Improve the ARIMA Model

```
from statsmodels.tsa.arima.model import ARIMA

model = ARIMA(series, order=(2, 4, 6))
model_fit = model.fit()
```

- 5.Print the model summary

```
print(model_fit.summary())
```

```
                                SARIMAX Results
==========================================================================================
Dep. Variable:     Mean monthly temperature   No. Observations:                  792
Model:                      ARIMA(2, 4, 6)   Log Likelihood               -2850.230
Date:                     Sat, 04 May 2024   AIC                           5718.460
Time:                             13:32:21   BIC                           5760.486
Sample:                         01-01-1907   HQIC                          5734.616
                              - 12-01-1972
Covariance Type:                      opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -1.9045      0.016   -117.984      0.000      -1.936      -1.873
ar.L2         -0.9961      0.015    -64.886      0.000      -1.026      -0.966
ma.L1         -0.3403      0.143     -2.386      0.017      -0.620      -0.061
ma.L2         -1.6235      0.101    -16.032      0.000      -1.822      -1.425
ma.L3          0.3404      0.137      2.492      0.013       0.073       0.608
ma.L4          0.7230      0.103      7.022      0.000       0.521       0.925
ma.L5         -0.2612      0.040     -6.585      0.000      -0.339      -0.183
ma.L6          0.1629      0.053      3.085      0.002       0.059       0.266
sigma2        79.2347     12.290      6.447      0.000      55.146     103.323
===================================================================================
Ljung-Box (L1) (Q):                   0.01   Jarque-Bera (JB):                11.33
Prob(Q):                              0.91   Prob(JB):                         0.00
Heteroskedasticity (H):               0.88   Skew:                             0.28
Prob(H) (two-sided):                  0.32   Kurtosis:                         3.14
===================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
```

- 6.Make a forecast (steps=10)

```
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from math import sqrt

# Splitting into train and test sets
train_size = int(len(series) * 0.75)
train_data, test_data = series[0:train_size], series[train_size:]

start = len(train_data)
end = len(train_data) + len(test_data) - 1
model = ARIMA(train_data, order=(9, 2, 0))
results = model.fit()
predictions = results.predict(start=start, end=end, dynamic=False, typ='levels')
forecast = results.predict(len(series), len(series) + 10, typ='levels')


print("Forecast (steps = 10)")
print(forecast)
```

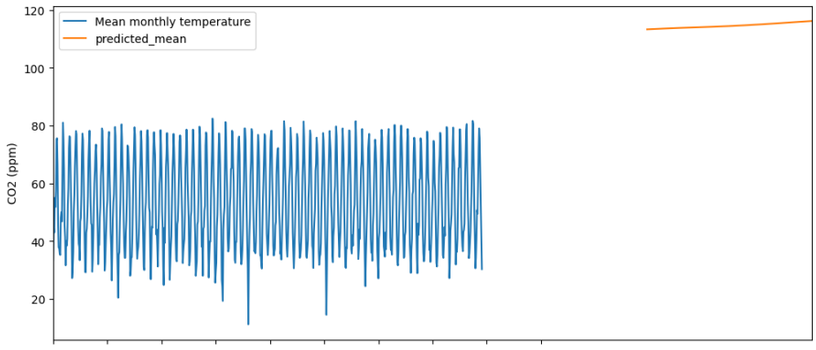
```
Forecast (steps = 10)
1973-01-01    113.284581
1973-02-01    113.551532
1973-03-01    113.772269
1973-04-01    113.970151
1973-05-01    114.173882
1973-06-01    114.409288
1973-07-01    114.692256
1973-08-01    115.024743
1973-09-01    115.394839
1973-10-01    115.780688
1973-11-01    116.156999
Freq: MS, Name: predicted_mean, dtype: float64
```

```
print("MSE: ", mean_squared_error(test_data, predictions))
print("RMSE: ", sqrt(mean_squared_error(test_data, predictions)))
```

```
MSE:  1360.2977048399682
RMSE:  36.88221393625887
```

- 7.Plot the forecast

```
import matplotlib.ticker as ticker
formatter = ticker.StrMethodFormatter('{x:,.0f}')


ylabel = 'CO2 (ppm)'
xlabel = 'Year'

ax = series.plot(legend=True, figsize=(12, 6))
forecast.plot(legend=True)
ax.autoscale(axis='x', tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter)
```



- 8.Perform a grid search

```
def evaluate_arima_model(dataset, arima_order):
  train_size = int(len(data) * 0.75)
  train_data, test_data = data[0:train_size], data[train_size:]

  start = len(train_data)
  end=len(train_data)+len(test_data)-1

  model = ARIMA(train_data, order = arima_order)
  results = model.fit()
  forecast = results.predict(len(data), len(data)+10, typ = 'levels')
  predictions = results.predict(start=start, end=end, dynamic=False, type='levels')
  mse = mean_squared_error(test_data, predictions)
  return mse


from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
import numpy as np

def evaluate_models(data, p_values, d_values, q_values):
    best_score, best_order = float("inf"), None
    for p in p_values:
        for d in d_values:
            for q in q_values:
                order = (p, d, q)
                try:
                    model = ARIMA(data, order=order)
                    model_fit = model.fit()
                    predictions = model_fit.predict()
                    mse = mean_squared_error(data, predictions)
                    if mse < best_score:
                        best_score, best_order = mse, order
                except:
                    continue
    print("Best ARIMA order:", best_order)
    print("Best MSE:", best_score)


import warnings
p_values = range(0, 11)
d_values = range(0, 4)
q_values = range(0, 4)
warnings.filterwarnings("ignore")
evaluate_models(series.values, p_values, d_values, q_values)

     Best ARIMA order: (10, 0, 3)
     Best MSE: 16.957872514692493


# Splitting into train and test sets
train_size = int(len(data) * 0.75)
train_data, test_data = data[0:train_size], data[train_size:]

start = len(train_data)
end=len(train_data)+len(test_data)-1
model = ARIMA(train_data, order = (9,2,0))
results = model.fit()
predictions = results.predict(start=start, end=end, dynamic=False, type='levels')
forecast = results.predict(len(data), len(data)+10, typ = 'levels')

ylabel='CO2(ppm)'
xlabel='Year'

ax = data.plot(legend=True,figsize=(12,6))
forecast.plot(legend=True)
ax.autoscale(axis='x',tight=True)
ax.set(xlabel=xlabel, ylabel=ylabel)
ax.yaxis.set_major_formatter(formatter);
```